DETERMINATION OF ONE OR MORE VARIABLES TO RECEIVE VALUE CHANGES IN LOCAL SEARCH SOLUTION OF INTEGER PROGRAMMING PROBLEM

Related Applications:

The following applications disclose related subject matter: U.S. Application No. (Attorney Docket No. 200209179-1), filed (on the same day as this application) and entitled, "Determining Placement of Distributed Application onto Distributed Resource Infrastructure"; and U.S. Application No. (Attorney Docket No. 200209180-1), filed (on the same day as this application) and entitled, "Incorporating Constraints and Preferences for Determining Placement of Distributed Application onto Distributed Resource Infrastructure"; the contents of all of which are hereby incorporated by reference.

Field of the Invention

The present invention relates to the field of solving an integer programming problem. More particularly, the present invention relates to the field of solving an integer programming problem where a constraint may include a polynomial term of at least second order.

Background of the Invention

An integer program can be used to model a resource allocation problem in which variables are assigned discrete values. An integer linear program expresses a particular resource allocation problem as a set of linear equations or inequalities. A method of solving an integer linear program employs a local search solution. The local search solution uses a gradient following approach to iteratively improve an initial assignment of values to the variables until a near optimum solution is reached. Each iteration of the local search solution produces a new assignment of values for the variables. Generally, the new assignment of values differs from a previous assignment of values by one value for a particular variable. For the integer linear program, a selection of the variables that follows the gradient is accomplished by evaluating coefficients for the variables in an unsatisfied constraint.

In a traditional integer program, the resource allocation problem is expressed in terms of constraints and an objective. The constraints are equations or inequalities. The objective is an optimization function.

Walser, in U.S. Patent No. 6,031,984, issued on Feb. 29, 2000, teaches a method of solving a linear integer program using a local search solution technique referred to as a WSAT(OIP) method. A linear program model for the WSAT(OIP) method defines the linear integer program as an overconstrained integer program, where hard constraints correspond to the constraints of the traditional integer program and where a sum of soft constraints corresponds to the objective. The WSAT(OIP) method selects the variables that follow the gradient by evaluating the coefficients for the variables in the unsatisfied constraint.

If the integer program includes polynomial terms of second or higher order (i.e., quadratic or higher order terms), the representation of coefficients of the variables as used in the WSAT(OIP) method are insufficient for an indication of the gradient.

What is needed is a method of representing and choosing a variable to receive a value change where a constraint includes a polynomial term of at least second order.

Summary of the Invention

The present invention is a method of determining a variable to receive a value change as part of a local search solution to an integer programming problem. The method can be used where a constraint has one or more polynomial terms of at least second order. In an embodiment of the present invention an unsatisfied constraint is selected. Stores are created for allowable changes of value for the variables in the unsatisfied constraint. The unsatisfied constraint is parsed through by term. For each variable in a term, the stores are updated with a change in the term for each of the allowable changes of the value while maintaining other variables constant. A variable to receive the value change, and possibly a value for the variable, are chosen based upon the store which meets at least one improvement criterion.

These and other aspects of the present invention are described in more detail herein.

Brief Description of the Drawings

The present invention is described with respect to particular exemplary embodiments thereof and reference is accordingly made to the drawings in which:

Figure 1 illustrates a preferred method of determining a variable to receive a value change as a block diagram according to an aspect of the present invention;

Figure 2 illustrates a first alternative method as a flow chart according to an aspect of the present invention, where the first alternative method employs a local search solution to solve an integer programming problem that includes a polynomial term of at least second order;

Figure 3 illustrates an alternative step of choosing a variable to receive a value change according to an aspect of the present invention; and

Figure 4 illustrates a system for determining a variable to receive a value change according to an aspect of the present invention.

Detailed Description of a Preferred Embodiment

The present invention is a method of choosing a variable to receive a value change as part of a local search solution to an integer programming problem. A constraint may include one or more polynomial terms of at least second order (i.e., a quadratic or higher order polynomial term). The local search solution uses a gradient following approach to iteratively improve an initial assignment of values to the variables until a satisfactory solution is reached. Each iteration of the local search solution produces a new assignment of values for the variables. Generally, the new assignment of values differs from a previous assignment of values by one value for a particular variable.

The integer programming problem may be modeled in a number of ways. For example, it may be modeled as a traditional integer programming problem or as an overconstrained integer programming problem. The traditional integer programming problem models a resource allocation problem in terms of constraints and an objective. The overconstrained integer programming problem models the resource allocation problem in terms of hard constraints and soft constraints. The hard constraints correspond to the constraints of the traditional integer programming problem. A sum of the soft constraints of the overconstrained integer programming problem corresponds to the objective of the traditional integer programming problem. The method of the present invention applies to the traditional integer programming

problem, the overconstrained integer programming problem, or another integer programming problem where a solver employs the local search solution to solve a problem model that includes the constraint having the polynomial term of at least second order.

A preferred method of the present invention is illustrated in Figure 1. The preferred method 100 determines a variable which is to receive a value change and a value for the variable. The preferred method 100 includes first through fourth steps, 102..108. In the first step 102, an unsatisfied constraint is selected. In the second step 104, stores are created for allowable changes of value for variables in the unsatisfied constraint. The stores are created for each variable in the unsatisfied constraint and for each allowable change of value of the variables. For example, if there are three variables in the unsatisfied constraint and if each variable has one allowable change of value, three of the stores are created.

In the third step 106, the unsatisfied constraint is parsed by term. For each variable in the term and for each allowable change of value of the variable, an associated store is updated with a change in the term while holding other variables constant. In the fourth step 108, the variable that receives the value change and the value for the variable is chosen according to at least one improvement criterion, such as the variable and the value which most improves the unsatisfied constraint.

In the present invention, the variables include Boolean variables, integer variables, and finite discrete value variables. The Boolean variables have only two discrete possible values such as {0, 1}. The integer variables have more than two discrete possible values such as {-5, 0, 5, 10, 20}. The finite discrete value variables include the integer variables and variables having discrete decimal values such as {1.3, 2.5, 8.9}. If a particular variable in the preferred method 100 is a Boolean variable, the allowable change to it is a single value. So in such a situation, the second step 104 would create a single store for the particular variable. Thus, if a Boolean variable is chosen in the fourth step 108, the value for the particular variable is found by flipping the previously assigned value.

If a second particular variable is an integer variable having ten possible values, the second step 104 would create nine stores for the second particular variable. Alternatively, the allowable changes of value in the second step 104 can be limited to a set of values near the previously assigned value for the integer variable. For example, if the integer variable is limited to values closest to the previously assigned

value, the second step 104 would create at most two stores for the integer variable. Or for example, if the integer variable is limited to two values immediately less than the previously assigned value and to two values immediately greater than the previously assigned value, the second step 104 would create at most four stores for the integer variable.

An example illustrates the preferred method 100. Given Boolean variables x and y having initial values of x = 0 and y = 1 and an unsatisfied constraint of

$$x^2 + 2xy + y^2 \geq 3,$$

the preferred method 100 begins in the first step 102 by selecting the unsatisfied constraint. The second step 104 then creates a first store for the variable x and a second store for the variable y each having an initial value of 0.

The third step 106 of parsing the unsatisfied constraint by term begins with the term of x^2, calculates that the change in the term for flipping the value of x is 1, and updates the first store so that the first store now has the value of 1. Since the variable y does not appear in the term of x^2, the value of the second store remains 0. The third step 106 continues with the term of 2xy. First the value of x is flipped while holding y constant. This results in a change of the term of 2xy of 2. The first store is updated with this change, which results in the first store having the value of 3. Next the value of y is flipped while holding x constant. This results in a change of the term of 2xy of 0. The second store may be updated with this change. As a result, the second store still has the value of 0. The third step 106 concludes with the term of y^2, calculates the change in the term for flipping the value of y is -1, and updates the second store so that the second store now has the value of -1. Since the variable x does not appear in the term of y^2, the value of the first store remains 3.

Thus, the first store concludes with the value of 3 and the second store concludes with the value of -1. This indicates that flipping the value of the first variable improves the unsatisfied constraint since the value of left-hand-side becomes 4 while flipping the value of the second variable does not improve the unsatisfied constraint as this would decrease the value of the left-hand-side to 0. The preferred method 100 then concludes by selecting the variable x as the variable to receive the value change since it is the only variable which can improve the unsatisfied constraint.

A first alternative method of the present invention is illustrated in Figure 2. The first alternative method 200 employs the local search solution to solve an

overconstrained integer programming problem. The first alternative method 200 includes the first through fourth steps, 102..108, of the preferred method 100, as well as fifth through ninth steps, 202..210. The first alternative method 200 begins with the fifth step 202, which defines a problem model having hard and soft constraints. In the sixth step 204, the variables are randomly assigned initial values. The preferred method 100 then determines the variable to receive the value change and the value for the variable.

In the seventh step 206, the assigned values are compared to optimality criteria to determine whether a solution has been found. The optimality criteria for the overconstrained integer programming problem are no violation of the hard constraints and near optimum solutions for the soft constraints. If the optimality criteria are not met, the first alternative method 200 continues in the eighth step 208 with a determination of whether an additional iteration is to be performed. If so, the first alternative method 200 returns to the preferred method 100 to determine another variable which is to be changed and a value for the variable. If not, a ninth step 210 determines whether to restart the first alternative method 200 by reinitializing the variables.. If the optimality criteria are met in the seventh step 206, a final value assignment for the variables is output as a result in a tenth step 212. If the ninth step 210 determines to not restart the first alternative method 200, a "no solution found" message is output in the tenth step 210.

A second alternative method of the present invention replaces the fourth step 108 in the first alternative method 200 with an alternative fourth step, which is illustrated in Figure 3. The alternative fourth step 300 includes first through fourth tasks, 302..308. In the first task 302, the stores are limited to available stores, which include the stores that improve the unsatisfied constraint and the stores that are not "tabu" stores. A tabu store is a store which indicates a variable and a value that was previously selected within a previously selected number of iterations, such as within ten iterations. By not choosing the tabu stores, the local search solution avoids getting stuck at a local optimum in its search for the near-optimum solution. The second task 304 chooses the available store that improves an overall solution at least as much as other available stores. This available store corresponds to the variable to receive the value change and the value for the variable. If no available store improves the overall solution, the third task 306 selects the store from the available stores by applying a probability p to a constraint improvement selection and a probability of (1 – p) to a

random selection. The constraint improvement selection chooses the variable and the value from the available stores that most improves the unsatisfied constraint. The random selection chooses a randomly selected variable and value from the available stores. In the fourth task 308, ties are broken according to a tie-breaking criterion, such as recency or frequency. Recency chooses the variable and the value based upon which variable and value was selected last longest ago. Frequency chooses the variable and the value based upon which variable and value have been chosen least often.

A fifth alternative method of the present invention modifies the third step 106 (Figure 1) of the preferred method 100. In the preferred method, the third step 106 parses the unsatisfied constraint by term. For each variable in the term and for each allowable change of value of the variable, the associated store is updated with the change in the term while holding other variables constant. In the fifth alternative method, for each variable in the term that is encountered for a first time in the parsing and for each allowable change of the value of the variable, the associated store is updated with a change in the unsatisfied constraint while holding the other variables constant. Updating the associated store with the change in the term of the preferred method 100 is preferred over updating the change in the unsatisfied constraint of the fifth alternative method because it minimizes term calculations for constraints where the variables appear in only a subset of the terms. For example, for a constraint of six terms and three variables where the three variables each appear in three terms, updating the stores for the change in the terms is accomplished in six term calculations while updating the stores for the change in the unsatisfied constraint is accomplished in eighteen term calculations. Minimizing the term calculations enhances an efficiency of the local search solution since the local search solution is an iterative technique where a total number of term calculations is a product of the term calculations per iteration and a number of iterations.

An embodiment of a system for employing the first alternative method of the present invention is illustrated schematically in Figure 4. The system 400 includes a display 402, input/output devices 404, a processing unit 406, a storage device 408, and a memory 410. The processing unit 406 couples to the display 402, the input/output devices 404, the storage device 408, and the memory 410.

In operation, the processing unit 406 reads the overconstrained integer programming problem into the memory 410. The processing unit 406 then initializes

the variables by randomly assigning values to the variables. Next, the processing unit 406 randomly selects the unsatisfied constraint. Following this, the processing unit 406 creates stores in the memory 410 for the allowable changes of the variables in the unsatisfied constraint. The processing unit 406 then parses the unsatisfied constraint by term updating individual stores associated with the term while maintaining other variables constant. Following this, the processing unit 406 chooses the variable to receive the value change and the value for the variable according to the improvement criterion. The processing unit 406 then determines whether the optimality condition has been met and, if not, determines whether to perform more iterations or whether the method should be restarted.

In an embodiment of the present invention, computer code resides on a computer readable memory, which is read into the system 400 by one of the input/output devices 404. Alternatively, the computer readable memory comprises the storage device 408 or the memory 410. The computer code provides instructions for the processing unit 406 to perform a method of the present invention. The computer readable memory is selected from a group including a disk, a tape, a memory chip, or other computer readable memory.

An exemplary set of constraints is provided in Table 1. The exemplary set of constraints includes first, second, and third constraints. The first and second constraints are categorized as hard constraints while the third constraint is categorized as a soft constraint.

Table 1

Constraint No.	Constraint	Type of Constraint
1	$x_1^2 + 2x_1x_2 + x_2^2 \geq 3$	Hard
2	$x_2^2 - 2x_2x_3 + x_3^2 \geq 15$	Hard
3	$x_1^2 + x_2^2 + x_3 \leq 11$	Soft

An exemplary set of variables is provided in Table 2. The exemplary set of variables includes first and second Boolean variables, x_1 and x_2, and an integer variable x_3.

Table 2

Variable	Type of Variable	Possible Values
x_1	Boolean	0, 1

x_2	Boolean	0, 1
x_3	Integer	0, 5, 10, 20

Applying the second alternative method to the exemplary constraints and the exemplary variables begins by randomly initializing the variables, for example $x_1 = 0$, $x_2 = 0$, and $x_3 = 20$. The step of selecting the unsatisfied constraint determines that the first and third constraints are unsatisfied constraints and randomly selects the first constraint. The step of creating stores in memory creates a first store for the first Boolean variable x_1 and creates a second store for the second Boolean variable x_2. The first and second stores are given initial values of 0.

The step of parsing through the first constraint by term begins with the term $(x_1)^2$, determines that the change in the term due to flipping $x_1 = 0$ with $x_1 = 1$ changes the value of the term by 1, and updates the first store with this change so that the first store now has a value of 1. The second store maintains the value of 0 since the second Boolean variable x_2 does not appear in the first term. The step of parsing through the first constraint by term continues with the term of $2x_1x_2$, determines that the change in the term due to flipping $x_1 = 0$ with $x_1 = 1$ changes the value of the term by 0, determines that the change in the term due to flipping $x_2 = 0$ with $x_2 = 1$ changes the value of the term by 0, and adds these changes to the first and second stores so that the first store has the value of 1 and the second store has the value of 0. The step of parsing through the first constraint by term concludes with the term of $(x_2)^2$, determines that the change in the term for flipping $x_2 = 0$ with $x_2 = 1$ changes the value of the term by 1, and adds this change to the second store so that the first store maintains the value of 1 and the second store concludes with the value of 1. Thus, flipping the value of either the first or second Boolean variables, x_1 or x_2, improves the unsatisfied constraint.

The step of choosing the variable and the value of the variable according to the second alternative method begins with the first task 302 (Figure 3) of identifying the first and second Boolean variable, x_1 and x_2, as available variables. It continues with a determination of whether the overall solution is improved and, if so, which of the available variables improves the overall solution at least as much as other available variables. To do this, a determination is first made of a sum of violations of the constraints with the assigned values of $x_1 = 0$, $x_2 = 0$, and $x_3 = 20$. The first constraint has a left hand side value of 0 with these values and, thus, a violation of the first constraint is 3. The second constraint has a left hand side value of 400 and, thus a

violation of the second constraint is 0 since a hard constraint cannot have a violation less than 0. The third constraint has a left hand side value of 20 and, thus, a violation of the third constraint is 9. Note that soft constraints can have a negative violation since, for the soft constraints, it is desirable to improve a solution beyond a given fixed value for the soft constraint. Thus, the sum of the violation of the constraints with the assigned values is 12.

Similarly, the sum of the violations of the constraints with the first Boolean variable flipped is determined to be 12 and the sum of the violations of the constraints with the second Boolean variable flipped is determined to be 12. Therefore, the second task 304 (Figure 3) of choosing the available store that improves the overall solution at least as much as the other available stores determines that flipping neither the first nor second Boolean variable improves the overall solution. The third task 306 (Figure 3) then determines that both the first and second Boolean variables improve the first constraint at least as much as the other and randomly selects the first Boolean variable to flip so that now $x_1 = 1$. Since the first and third constraints remain unsatisfied constraints, the seventh step 206 (Figure 2) of determining whether the assigned values of $x_1 = 1$, $x_2 = 0$, and $x_3 = 20$ meets the optimality criteria determines that they do not. The eighth step 208 then determines that an additional iteration is to be performed.

The method continues with selection of the third constraint as the unsatisfied constraint, determination that the variable to receive the value change is the integer variable, and determination that the value for the variable is 5. In this iteration the sum of the violations the constraints is -3. Also, since the first constraint continues to be unsatisfied, the optimality condition is not met in the seventh step 206. The eighth step 208 then determines that an additional iteration is to be performed. In a final iteration, the first constraint is selected since it is the only remaining unsatisfied constraint. The final iteration determines that the second Boolean variable, x_2, is to be flipped and outputs a result of $x_1 = 1$, $x_2 = 1$, and $x_3 = 5$.

It will be readily apparent to one skilled in the art that application of the second alternative method to the exemplary constraints and variables is meant to illustrate an embodiment of the present invention and does not represent a typical application. A more typical application would involve many more constraints and many more variables. The more typical application is only limited in size by

processing speed and power of a system implementing the solution and by a desired time for determining a result.

The foregoing detailed description of the present invention is provided for the purposes of illustration and is not intended to be exhaustive or to limit the invention to the embodiments disclosed. Accordingly, the scope of the present invention is defined by the appended claims.